Technical Report - Project specifications

InOutTracker

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 25-01-2022
Students:	98188: Daniel Francisco 98324: Henrique Sousa 98498 : Daniel Figueiredo 100244: Rui Júnior
Project abstract:	Tracker to manage public spaces capacity, entrances and exits during the pandemic and in general. It will be helpful for mall customers, security guards, managers and analysts

Table of contents:

1 Introduction

2 Product concept
- Vision statement
- Personas
- Main scenarios

3 Architecture notebook
- Key requirements and constrains
- Architetural view
- Module interactions

4 Information perspetive

5 References and resources

1 Introduction

The world has been dealing with a pandemic for the past two years. This has led to some restrictions, namely, the maximum capacity of people in public places. Mechanisms for managing these limits are necessary, a small store with a limit of few people easily controls this manually, but when we are dealing with a shopping center, for example, where there are several different entrances and exits it is impossible to control how many people are in there at any given time so it requires additional technical help.

In a world without pandemic restrictions, there is also continuous waiting in lines at stores, so it would be useful to have a system where we could access the capacity of a public place in order to decide whether we intend to go there or not.

Our system seeks to help in the management of shopping centers and stores, but also in providing the information acquired by the entrance and exit sensors to customers so that they have access to the current capacity of a mall and/or store.

2 Product concept

Vision statement.

The system will manage all the entrances and exits of stores in several malls. It will be possible for people to check whether it is possible to enter a store or if the capacity of the store is at the maximum, being the main goal of the system to control the agglomerations of people due to the pandemic.

With this, we pretend that agglomerations can be controlled inside malls and/or stores, but also for clients to see if the mall or specific stores are overcrowded so they don’t have to wait if that happens.

In case there are agglomerations inside the malls or stores, the securities of the mall will have their life easen because with the application it will be easier for them to know where the agglomerations are, and they can go there and solve this issue.

The application will also have a manager side, that will allow for mall managers to see the statistics of every store, like how many people go there in a day

With this being said our application works for managers, security guards and clients.

Personas

Name: *Bruna*

Gender: *Female*

Age: *40*

Profession: *Lawyer*

Background: *Bruna likes to go shopping*

Problem: *Sometimes the shops are full of people and since Covid started they have a limit of people that can enter the store and the waiting time is to much for her*

Needs: *An app to check how busy the mall and stores are*

Bruna is a forty years old Lawyer who obtained her degree eleven years ago at University of Coimbra. She is married and has two kids, one with seven years old and a sixteen years old. **Bruna** is very busy since her job makes her change places very often. She likes to go shopping in her free time and buy gifts for her children. However, one thing she doesn't like is to wait in line to go to a store, so every time she thinks of going shopping she wonders if there's a lot of people there.

MOTIVATION: **Bruna** would like to check how busy a mall or store is before leaving home and wasting time waiting in line.

Name: *Pedro*

Gender: *Male*

Age: *38*

Profession: *Security Guard*

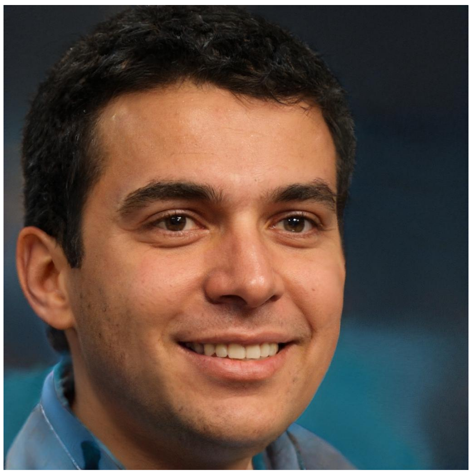
Background: *Pedro makes sure the mall and stores are secure and the DGS security measures are being respected*

Problem: *A mall has multiple stores and is too big for him to manage it without any technical help*

Needs: *A Software that manages the mall and notifies him of potential incorrect/dangerous behaviors*

Pedro is a thirty-eight years old Security Guard that has worked at Fórum Aveiro since he was thirty years old. He is single and has no children. Since **Pedro** started working at Fórum Aveiro the amount of work has increased overtime along with the movement at the mall. Although he is not alone guarding the mall he feels like it is too big and has too many stores to keep track at the same time now that the mall requires a minimum number of people. However, he has to watch the security cameras manually and has no information of how many people are in a certain place in the mall

MOTIVATION: **Pedro** needs a software that can manage the number of people in each place and notifies him (if needed) to disperse some group of people or maintain order in the stores



Name: *Manuel*

Gender: *Male*

Age: *48*

Profession: *Manager*

Background: *Manuel is the manager and analyst of some malls*

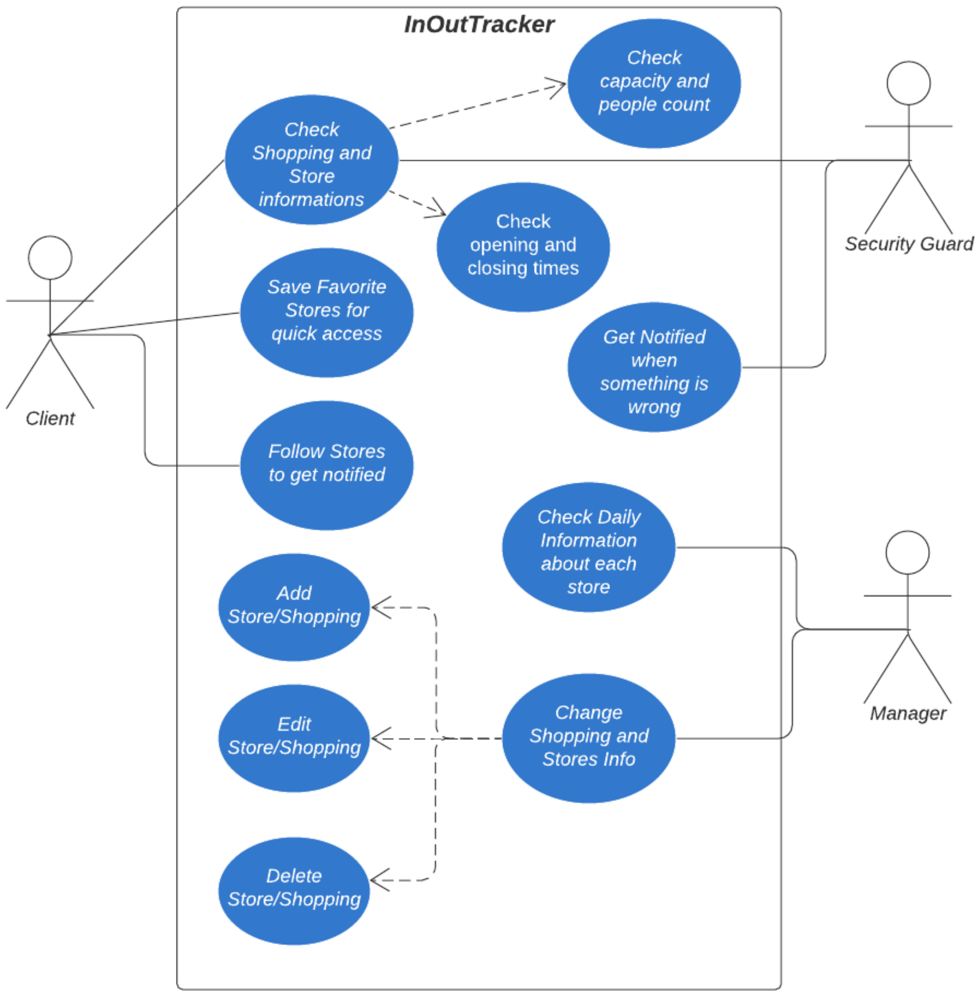
Problem: *Its difficult to analyze a store performance without some official values captured by a real time software*

Needs: *A Software that manages the mall and stores the relevant information for further analysis*

Manuel is a forty-eight years old Analyst who currently works for a few malls located in Aveiro. He is married and has two children. Since **Manuel** started working for these malls he found a lack of software to help him do his job more accurately and easier.

MOTIVATION: **Manuel** needs an accurate and real time software that can manage the number of people in each place and compile every information regarding the number of people each day.

Main scenarios


InOutTracker
Check capacity and people count
Check Shopping and Store informations
Check opening and closing times
Save Favorite Stores for quick access
Get Notified when something is wrong
Follow Stores to get notified
Check Daily Information about each store
Add Store/Shopping
Edit Store/Shopping
Change Shopping and Stores Info
Delete Store/Shopping
Client
Security Guard
Manager

Bruna checks mall capacity - Bruna opens the application and chooses the desired mall/stores that she wishes to go shopping. She sees that the mall is not too busy and there's close to no one in the store she wants to go to, so she leaves the house and goes shopping knowing that she won't lose time waiting in line.

Pedro gets notified that there's more people than allowed at a store - Pedro has the software running and connected to the mall surveillance system. A group of people starts to form at a store and there's more people than allowed in that store. The software alerts Pedro of the situation and he immediately goes there to solve the issue.

Manuel is asked to analyze last week's performance of a store - Manuel logs in with his manager account and chooses the store he wishes to fetch the data. He will be presented with a graph with how many people were in that store for each hour of the day.

3 Architecture notebook

Key requirements and constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- The system needs to be able to generate data automatically, simulating what happens in real malls.
- The system needs to be capable of keeping up with all the entrances and exits of people in all malls and stores.
- The application needs to be always consistent and updated, so that people (clients and security guards) that use the app don't be misled by the real number of people in a certain place.
- At the end of each virtual day, the system needs to be able to update all the statistics for managers.

Architectural view

Our service has a MySQL database and a pulsar standalone client running in the background. After these two components are running we start the SpringBoot API which will connect to the Database and create a pulsar consumer. After that, we start the data generation python script that simulates real life flow for some shoppings and their stores. In this script we have to connect to the api and pulsar client. There's a producer for the message broker and in this script we generate a few data just for demonstration purposes. It periodically sends messages to the message broker (pulsar) through the created producer. The API consumer receives these messages and updates the database with the new information that later will be accessed via mobile and web apps that get database info from the API.

We have a mobile app because it made sense for our product. As we want our service to be easily accessible and to provide real time information it is important, especially for shopping clients that they can access the information in a mobile format. That said we focused our mobile app for the clients and the web app for security guards and analysts of the malls.

Web application : To build the application we choose to use HTML Native as well as JavaScript. With JS it will be possible to make requests to the API.

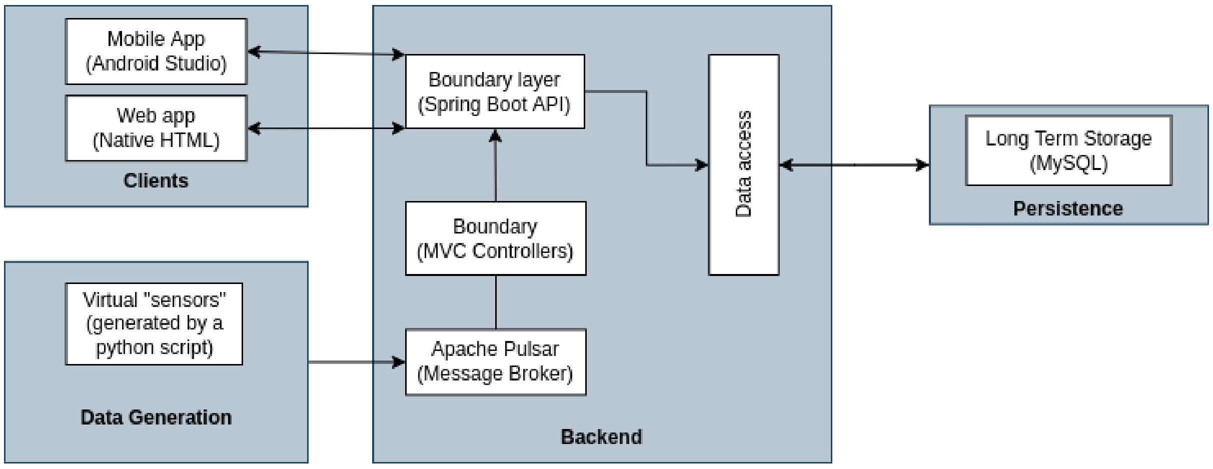
Mobile App: To build our mobile application we choose Android Studio that periodically fetches data through GET requests to the API.

Services : Service based on *Spring Boot*, which will be the essential point of the system.

Data simulation : Generate data and simulate a shopping flow through a python script.

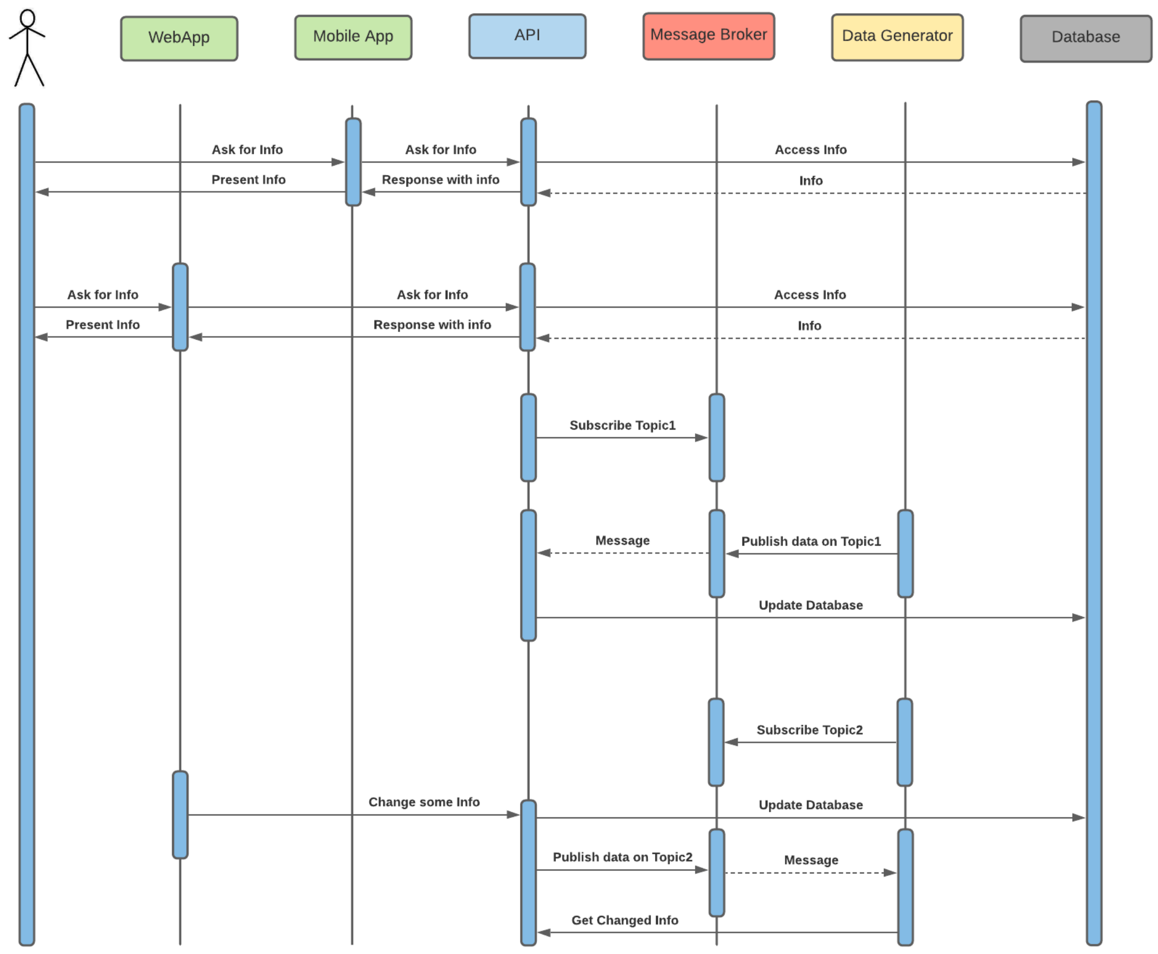
Message Broker : We choose Apache Pulsar as our message broker. With these we pretend to update data that comes from the data generator so we can keep the system updated.

Database : MySQL to store relational data from the system.
Although our service's main objective doesn't require a persistent database, since the past data is not relevant to the present user, we will have a persistent database in order to help managers and analysts to get information, like how many people enter a store in a day or how much time someone spends at a specific store. With that done our service will help clients, security guards and managers with the same resources.


Mobile App (Android Studio)
Web app (Native HTML)
Clients
Virtual "sensors" (generated by a python script)
Data Generation
Boundary layer (Spring Boot API)
Boundary (MVC Controllers)
Apache Pulsar (Message Broker)
Data access
Backend
Long Term Storage (MySQL)
Persistence
Figure 1: InOutTracker Architecture

Module interactions

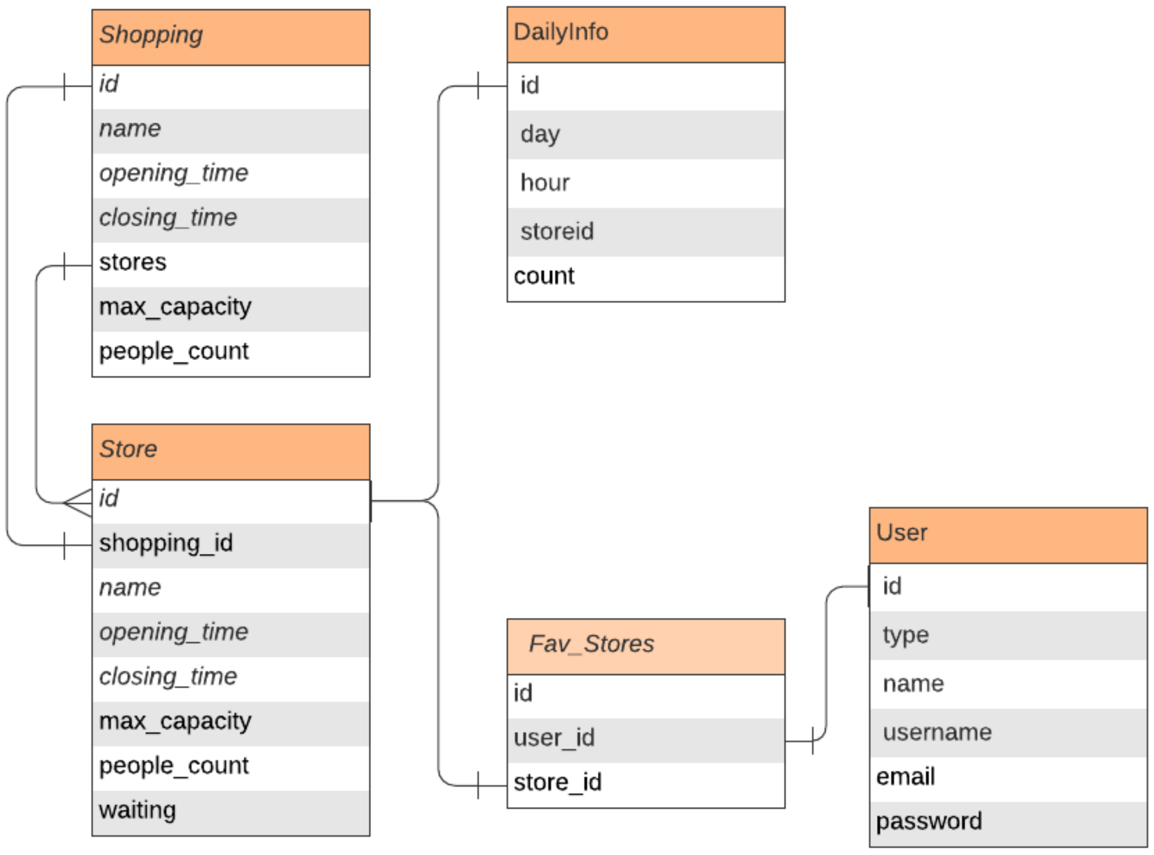
1. The data will be generated through python scripts, and then published in a topic in the message broker.
2. The data in the message broker will be consumed by the logic implemented in *Spring Boot,* storing the data in the database.
3. Through the connection of the application with the back-end it will be possible for the application to receive processed data.
4. The client sees the information about a store, being made a request to the API for this purpose.
5. Through the API the requested data is sended to the database and then passed this information to the message broker.
6. Then the message broker will connect to the python script, which will generate new data.



4 Information perspective

Being the entities related to each other the most viable option was to use a relational database, in this case, we choose MySQL.

The modulation of the database is structured in the following way:



5 References and resources

https://pulsar.apache.org/docs/en/concepts-overview/
https://www.geeksforgeeks.org/how-to-create-a-rest-api-using-java-spring-boot/
https://dev.mysql.com/doc/mysql-getting-started/en/
https://developer.android.com/studio/intro